1. **Divide by zero exception.**

Write a program to obtain two numbers and print their quotient. In case of exception print the same.

1. Write a program to read the Register Number and Mobile Number of a student. Create user defined exception and handle the following:
2. If the Register Number does not contain exactly 9 characters in specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException.
3. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.
4. If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.
5. If they are valid, print the message ‘valid’ else ‘Invalid’.
6. Write a program to valid the email address and display suitable exception if there is any mistake.

Create 3 custom exceptions class as below

1. **DotException**
2. **AtTheRateException**
3. **DomainException**

A typical email address should have a **" . "** character, **"@"** character and also the domain name should be valid. Valid domain names for practice be '**in**', '**com**', '**net**' or '**biz**'.

Display **Invalid Dot usage**, **Invalid @ usage** or **Invalid Domain** message based on email id.

Get the email address from the user, validate the email by checking the above-mentioned criteria and print the validity status of the input email address.

1. **NumberFormatException**

Another common type of exception which you would have come across already. When you use BufferedReader to read input you need to parse String it into various datatype like Integer, Double. For example, If you try to parse a String ("abc") into Integer, it throws NumberFormatException. So let's try to handle this NumberFormat exception.

In our application, while acquiring attributes for classes like ItemType, this exception may occur. So try to handle it in this program.

Create a class ItemType with the following attribute,

Add appropriate getter/setter, default and parameterized constructor. public ItemType(String name, Double deposit, Double costPerDay). Override toString() and print the details.

Handle the NumberFormatException in the Main Class.

Refer sample input/output for other further details and format of the output.

1. **NullPointerException**

Another prominent exception is NullPointerException. It occurs when you try to access a null value. Assign null value to a string and obtain an index position and try to access it. Print the exception.

1. **User defined Exception**

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, user can also create exceptions which are called ‘user-defined Exceptions’.

Create a class **Bank** with the following private attributes and Create class BankBO with the following method.

Include appropriate getters/setters and add constructors.

Create a driver class called Main. In the Main method, obtain inputs from the user. Validate the balance and if there is an exception, handle the exception and prompt the user(Refer I/O)

Pass the exception message as "Balance is less than 1000".

1. Mohan, a librarian is creating software to automate his work. A part of this, he needs to handle the exception if the purchase quantity is greater than the available quantity.

Create an Exception class called **BookQuantityNotAvailableException**and use it in the class called **“Book”** which is described by its book Id, book title, author, price and quantity available. Include a method called purchase() taking the purchased quantity as a parameter and update the quantity available appropriately. Print suitable exception if the purchased quantity is more than the available quantity. Help Mohan to complete this task. Refer sample input and output.

1. You are required to compute the power of a number by implementing a calculator. Create a class **MyCalculator** which consists of a single method **long power(int, int)**. This method takes two integers, **n** and **p**, as parameters and finds np. If either n or p is negative, then the method must throw an Exception which says "**n and p should be non-negative**". Also, if both n and p are zero, then the method must throw an Exception which says “ **n and p should not be zero**”. Complete the function power in class MyCalculator and return the appropriate result after the power operation or an appropriate exception as detailed above.